NAME

add_wch, **wadd_wch**, **mvadd_wch**, **mvwadd_wch**, **echo_wchar**, **wecho_wchar** − add a complex character and rendition to a **curses** window, then advance the cursor

SYNOPSIS

#include <curses.h>

int add_wch(const cchar_t ****wch* **);
int wadd_wch(WINDOW ****win*, const cchar_t ****wch* **);**
int mvadd_wch(int *y***, int** *x***, const cchar_t *****wch* **);**
int mvwadd_wch(WINDOW *****win*, int** *y***, int** *x***, const cchar_t *****wch* **);**
int echo_wchar(const cchar_t *****wch* **);
int wecho_wchar(WINDOW *****win*, const cchar_t *****wch* **);**

DESCRIPTION

add_wch

The **add_wch**, **wadd_wch**, **mvadd_wch**, and **mvwadd_wch** functions put the complex character *wch* into the given window at its current position, which is then advanced. These functions perform wrapping and special-character processing as follows:

- If *wch* refers to a spacing character, then any previous character at that location is removed. A new character specified by *wch* is placed at that location with rendition specified by *wch*. The cursor then advances to the next spacing character on the screen.
- If *wch* refers to a non-spacing character, all previous characters at that location are preserved. The non-spacing characters of *wch* are added to the spacing complex character, and the rendition specified by *wch* is ignored.
- If the character part of *wch* is a tab, newline, backspace or other control character, the window is updated and the cursor moves as if **addch** were called.

echo_wchar

The **echo_wchar** function is functionally equivalent to a call to **add_wch** followed by a call to **refresh**. Similarly, the **wecho_wchar** is functionally equivalent to a call to **wadd_wch** followed by a call to **wrefresh**. The knowledge that only a single character is being output is taken into consideration and, for non-control characters, a considerable performance gain might be seen by using the ***echo*** functions instead of their equivalents.

Line Graphics

Like **addch**(3X), **addch_wch** accepts symbols which make it simple to draw lines and other frequently used special characters. These symbols correspond to the same VT100 line-drawing set as **addch**(3X).

Name	*Unicode*	*Default*	*Description*
WACS_BLOCK	0x25ae	#	solid square block
WACS_BOARD	0x2592	#	board of squares
WACS_BTEE	0x2534	+	bottom tee
WACS_BULLET	0x00b7	o	bullet
WACS_CKBOARD	0x2592	:	checker board (stipple)
WACS_DARROW	0x2193	v	arrow pointing down
WACS_DEGREE	0x00b0	’	degree symbol
WACS_DIAMOND	0x25c6	+	diamond
WACS_GEQUAL	0x2265	>	greater-than-or-equal-to
WACS_HLINE	0x2500	−	horizontal line
WACS_LANTERN	0x2603	#	lantern symbol
WACS_LARROW	0x2190	<	arrow pointing left
WACS_LEQUAL	0x2264	<	less-than-or-equal-to
WACS_LLCORNER	0x2514	+	lower left-hand corner
WACS_LRCORNER	0x2518	+	lower right-hand corner
WACS_LTEE	0x2524	+	left tee
WACS_NEQUAL	0x2260	!	not-equal
WACS_PI	0x03c0	*	greek pi
WACS_PLMINUS	0x00b1	#	plus/minus

WACS_PLUS	0x253c	+	plus
WACS_RARROW	0x2192	>	arrow pointing right
WACS_RTEE	0x251c	+	right tee
WACS_S1	0x23ba	−	scan line 1
WACS_S3	0x23bb	−	scan line 3
WACS_S7	0x23bc	−	scan line 7
WACS_S9	0x23bd	_	scan line 9
WACS_STERLING	0x00a3	f	pound-sterling symbol
WACS_TTEE	0x252c	+	top tee
WACS_UARROW	0x2191	^	arrow pointing up
WACS_ULCORNER	0x250c	+	upper left-hand corner
WACS_URCORNER	0x2510	+	upper right-hand corner
WACS_VLINE	0x2502	|	vertical line

The wide-character configuration of ncurses also defines symbols for thick- and double-lines:

Name	*Unicode*	*Default*	*Description*
WACS_T_ULCORNER	0x250f	+	thick upper left corner
WACS_T_LLCORNER	0x2517	+	thick lower left corner
WACS_T_URCORNER	0x2513	+	thick upper right corner
WACS_T_LRCORNER	0x251b	+	thick lower right corner
WACS_T_LTEE	0x252b	+	thick tee pointing right
WACS_T_RTEE	0x2523	+	thick tee pointing left
WACS_T_BTEE	0x253b	+	thick tee pointing up
WACS_T_TTEE	0x2533	+	thick tee pointing down
WACS_T_HLINE	0x2501	-	thick horizontal line
WACS_T_VLINE	0x2503	|	thick vertical line
WACS_T_PLUS	0x254b	+	thick large plus or crossover
WACS_D_ULCORNER	0x2554	+	double upper left corner
WACS_D_LLCORNER	0x255a	+	double lower left corner
WACS_D_URCORNER	0x2557	+	double upper right corner
WACS_D_LRCORNER	0x255d	+	double lower right corner
WACS_D_RTEE	0x2563	+	double tee pointing left
WACS_D_LTEE	0x2560	+	double tee pointing right
WACS_D_BTEE	0x2569	+	double tee pointing up
WACS_D_TTEE	0x2566	+	double tee pointing down
WACS_D_HLINE	0x2550	-	double horizontal line
WACS_D_VLINE	0x2551	|	double vertical line
WACS_D_PLUS	0x256c	+	double large plus or crossover

RETURN VALUE

All routines return the integer **ERR** upon failure and **OK** on success.

Functions with a "mv" prefix first perform a cursor movement using **wmove**, and return an error if the position is outside the window, or if the window pointer is null.

NOTES

Note that **add_wch**, **mvadd_wch**, **mvwadd_wch**, and **echo_wchar** may be macros.

PORTABILITY

All of these functions are described in the XSI Curses standard, Issue 4. The defaults specified for line-drawing characters apply in the POSIX locale.

X/Open Curses makes it clear that the WACS_ symbols should be defined as a pointer to **cchar_t** data, e.g., in the discussion of **border_set**. A few implementations are problematic:

- NetBSD curses defines the symbols as a **wchar_t** within a **cchar_t**.
- HPUX curses equates some of the *ACS_* symbols to the analogous *WACS_* symbols as if the *ACS_* symbols were wide characters. The misdefined symbols are the arrows and other symbols which are not used for line-drawing.

X/Open Curses does not define symbols for thick- or double-lines. SVr4 curses implementations defined their line-drawing symbols in terms of intermediate symbols. This implementation extends

those symbols, providing new definitions which are not in the SVr4 implementations.

SEE ALSO

ncurses(3NCURSES), **addch**(3NCURSES), **attr**(3NCURSES), **clear**(3NCURSES), **outopts**(3NCURSES), **refresh**(3NCURSES), **putwc**(3)